NSCS

المدرسة الوطنية العليا في الأمن السيبراني

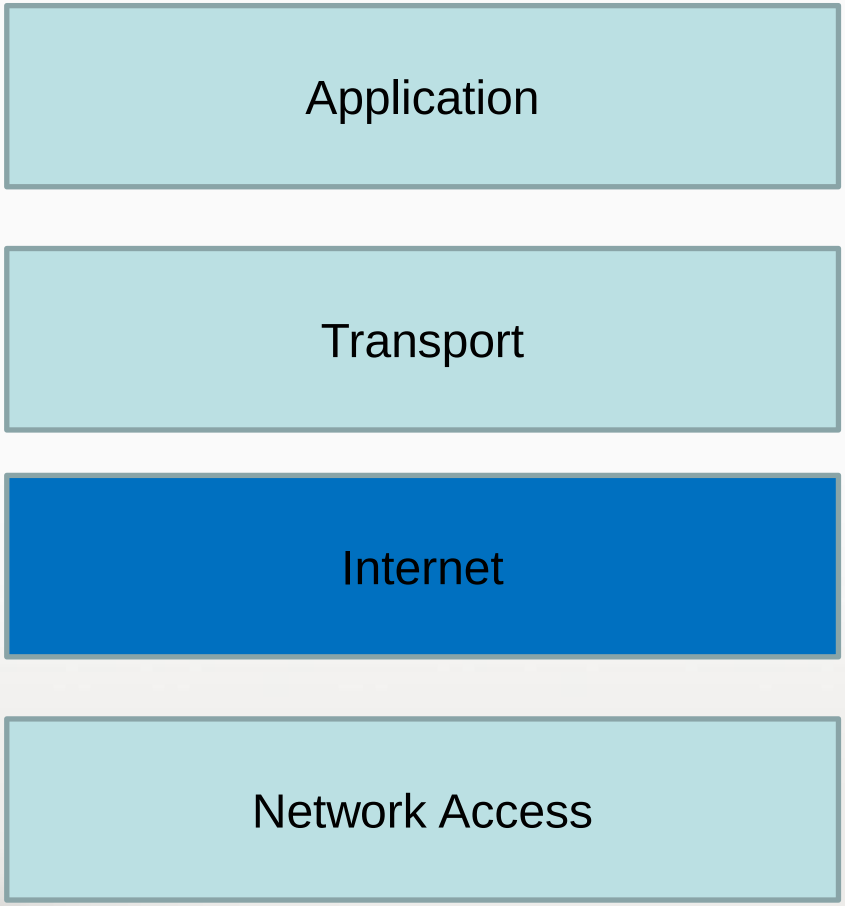
NATIONAL SCHOOL OF CYBERSECURITY

Transport Layer

The transport layer is responsible for the end-to-end communication between devices across a network. It ensures reliable data transmission, error correction, and data flow control. This layer establishes a logical connection between the source and destination, ensuring that data is delivered accurately and in the correct order.

By Pr RIAHLA Mohamed Amine

Study of TCP/IP Layers: Internet Layer



Application

Transport

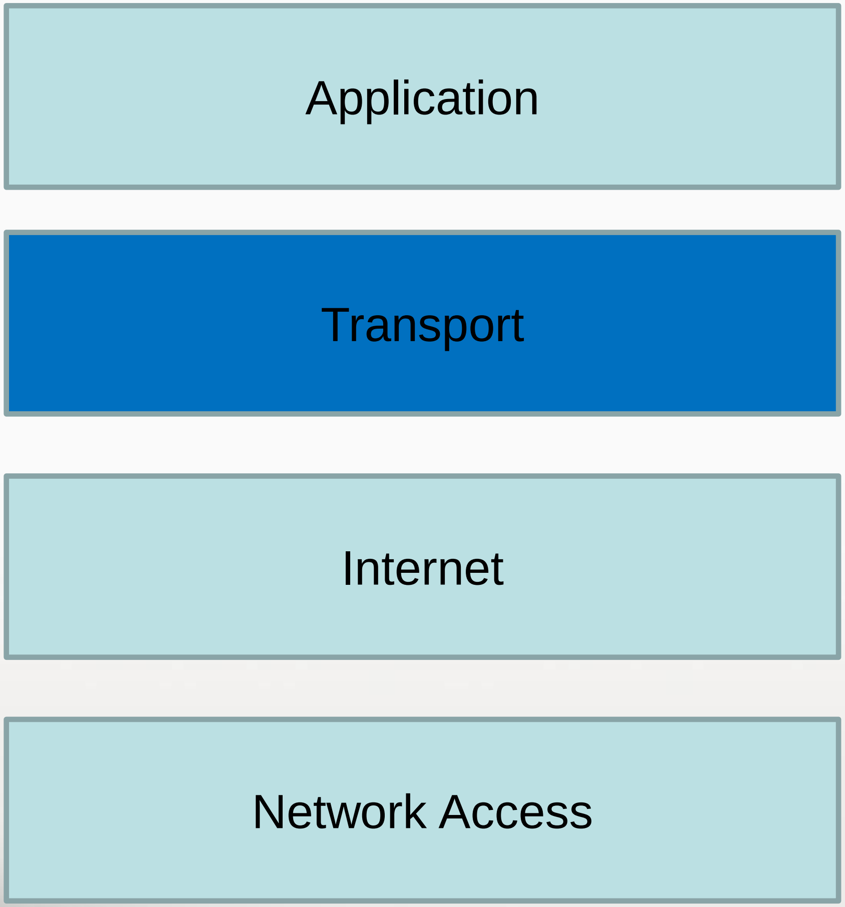
Internet

Network Access

Addressing and Routing

Datagram or IP Protocol
ARP Protocol
ICMP Protocol

Study of TCP/IP Layers: Transport Layer


Application

Concept of Ports

Transport

TCP Protocol
UDP Protocole

Internet

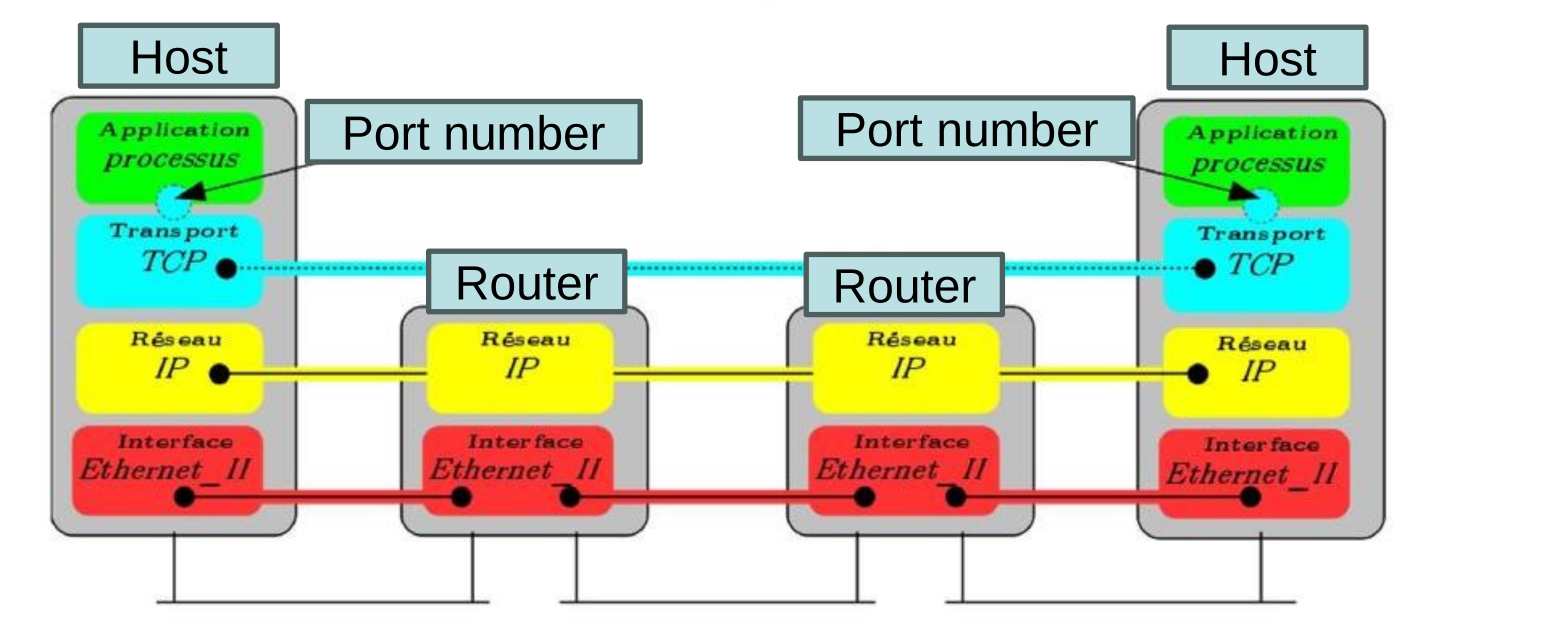
Network Access

Transport Layer and Ports

Ports are used to distinguish between different applications and processes on a device. Each port number corresponds to a specific service or application. Ports allow multiple services (such as a web server, email server, or file server) to communicate simultaneously without interfering with each other.

Transport Layer and Ports

- The previous layers allow information to be sent from one machine to another.
- The transport layer allows applications running on remote machines to communicate



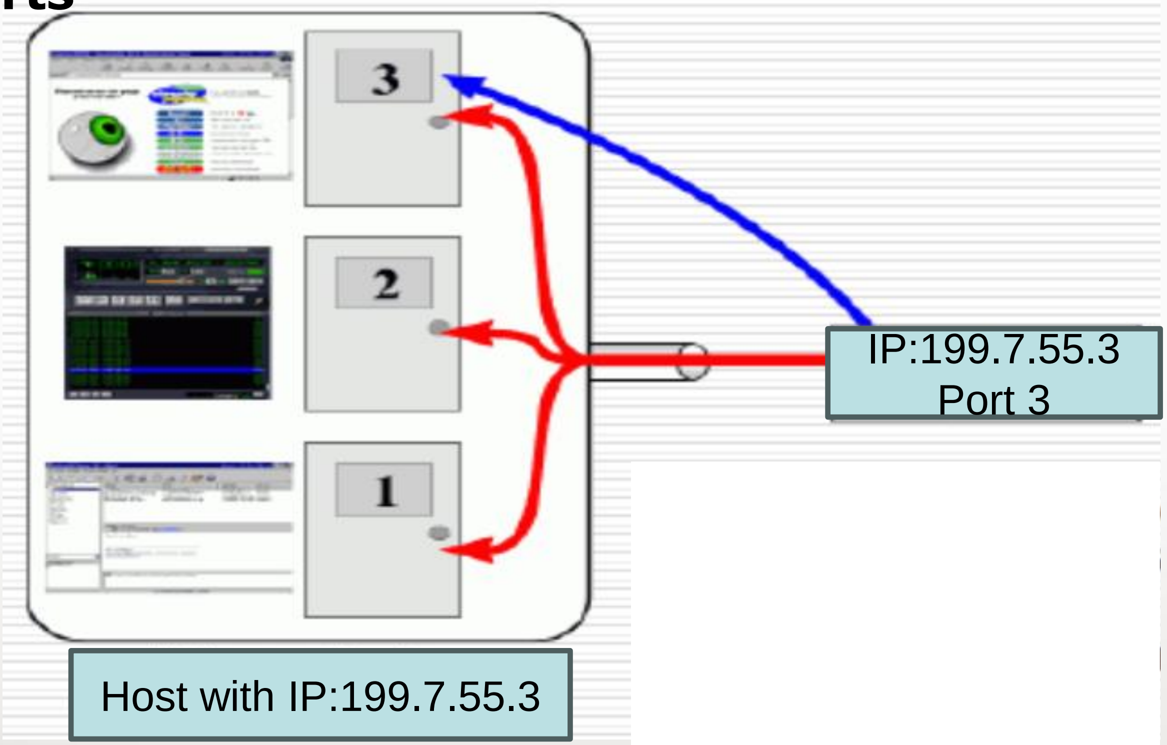
Transport Layer Introduction

- The role of the internet layer stops when the source machine reaches the destination (using routing, addressing, etc.).
- Between two machines, it’s mostly applications that communicate (Messenger, Viber, Instagram, etc.)
- The name of these applications varies from one system to another.

The problem is to identify these applications.

Transport Layer
Solution: port

- A numbering system for each application has been implemented: **ports**



- This port number is written in 2 bytes, which gives 65535 possible ports.

- **1-1024**: reserved Ports
- **1025-65535:** free Ports

Transport Layer
Solution: port

Well-Known Ports (0 to 1023)

These are reserved for popular services and protocols that are widely recognized.

- **Port 80**: HTTP (HyperText Transfer Protocol), used for web browsing.
- **Port 443**: HTTPS (HyperText Transfer Protocol Secure), used for secure web browsing.
- **Port 25**: SMTP (Simple Mail Transfer Protocol), used for sending emails.
- **Port 21**: FTP (File Transfer Protocol), used for file transfers.

Registered Ports (1024 to 49151)

These ports are assigned by the Internet Assigned Numbers Authority (IANA) for specific applications that are not as widely used as well-known services. Examples include:

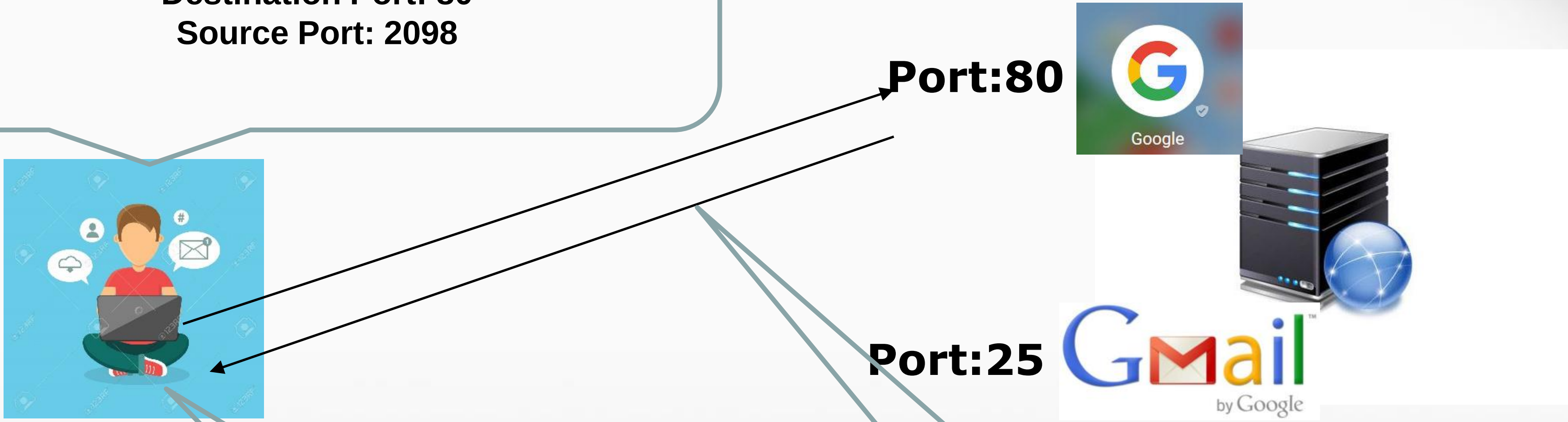
- **Port 3306**: MySQL database service.
- **Port 5432**: PostgreSQL database service.
- **Port 8080**: used for HTTP-based web services, often for testing or development.

Dynamic or Private Ports (49152 to 65535)

These are temporary ports used for ephemeral (short-lived) connections. These ports are usually assigned dynamically by the operating system for client applications when initiating communication with a server. Once the connection is closed, the port is released for use by other applications.

Transport Layer
Solution: port

I want to search on Google, so:
Destination Port: 80
Source Port: 2098



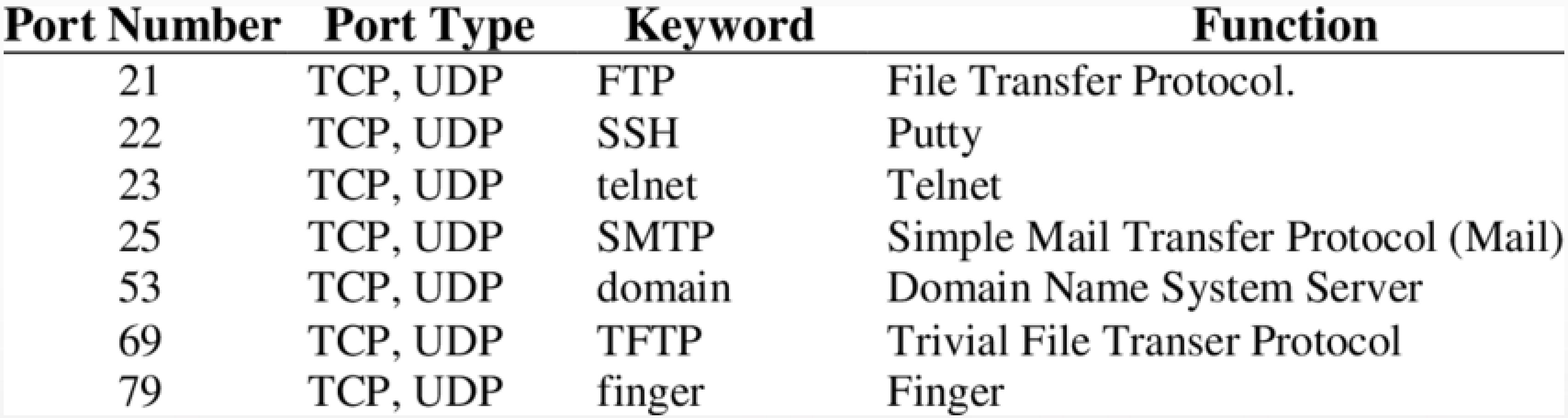
Port:80

Port:25

Next time, I will send an email, so I will contact port 25

Google's response:
Source Port: 80
Destination Port: 2098

Transport Layer
Solution: port



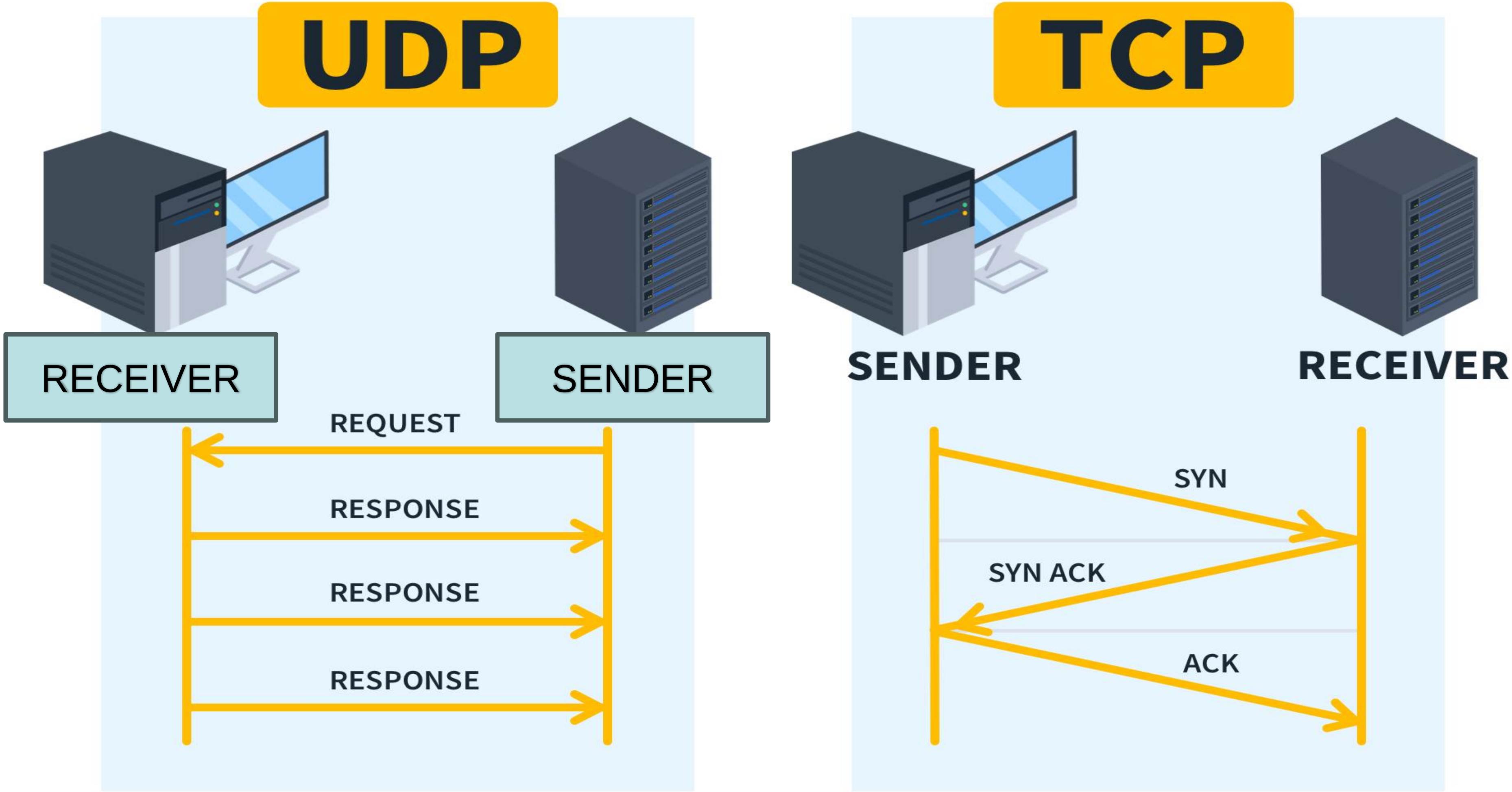
Port Number	Port Type	Keyword	Function
21	TCP, UDP	FTP	File Transfer Protocol.
22	TCP, UDP	SSH	Putty
23	TCP, UDP	telnet	Telnet
25	TCP, UDP	SMTP	Simple Mail Transfer Protocol (Mail)
53	TCP, UDP	domain	Domain Name System Server
69	TCP, UDP	TFTP	Trivial File Transer Protocol
79	TCP, UDP	finger	Finger

It’s routing to applications.

Transport Layer Protocols

By Pr RIAHLA Mohamed Amine

Transport Layer Protocols



Le protocole UDP
(User Datagram Protocol)

By Pr RIAHLA Mohamed Amine

Key Features of UDP:

- **Connectionless**: UDP does not establish a connection before sending data. It simply sends packets (datagrams) to the destination without checking if the receiver is ready. This makes it faster but less reliable compared to TCP.

- **Unreliable**: There is no acknowledgment of data receipt, and there is no retransmission in case of lost packets. If packets are lost or corrupted during transmission, they are not recovered.

- **Lightweight**: UDP has a smaller header size (8 bytes) compared to TCP (20 bytes). This makes it more efficient for sending small amounts of data or for applications that don't require a reliable connection.

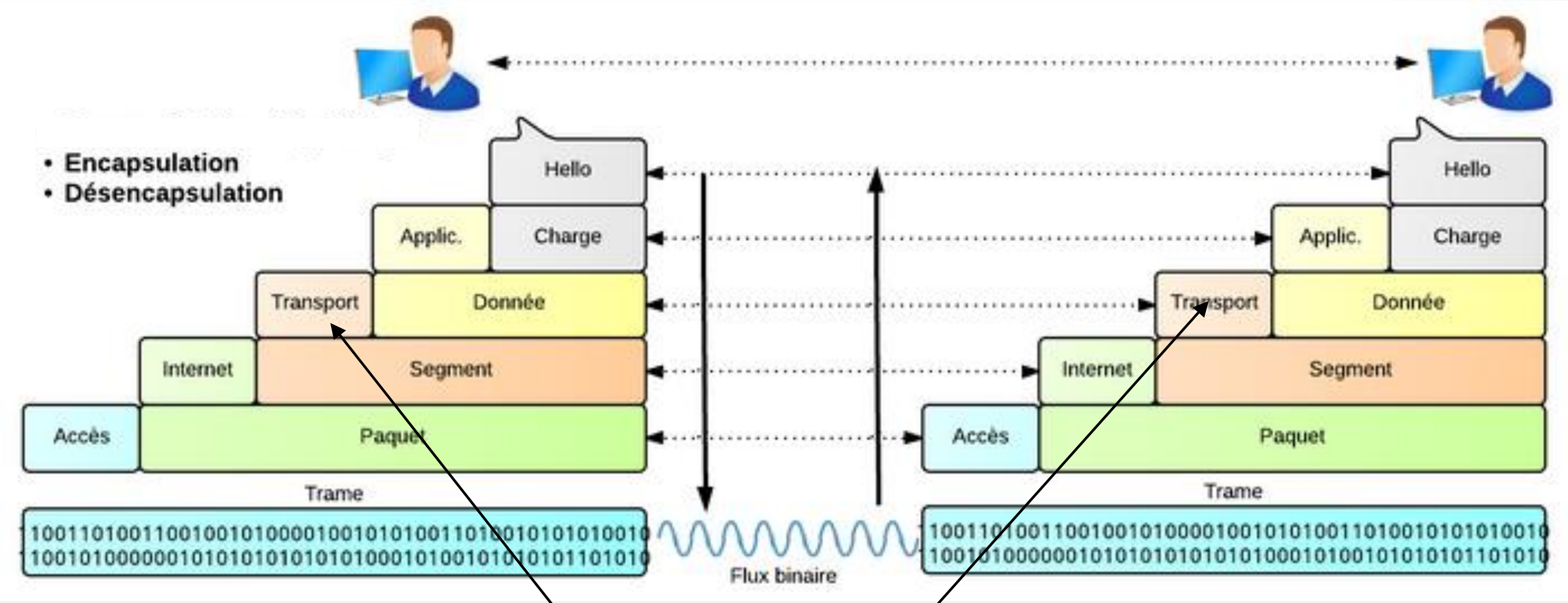
Key Features of UDP:

- **No Flow Control**: Unlike TCP, UDP does not have flow control mechanisms. The sender can send data as fast as it wants, and it is up to the application or the network to handle any issues.
- **Order of Delivery**: UDP does not guarantee that packets will arrive in the same order they were sent. If order is important, it must be handled by the application layer.
- **Error Checking**: UDP includes a checksum for error checking, which helps detect if data has been corrupted during transmission. However, there is no mechanism for automatic correction or retransmission.

Common Use Cases

- **Streaming**: Video or audio streaming services often use UDP because they prioritize speed and can tolerate some data loss.
- **DNS (Domain Name System)**: The DNS protocol uses UDP because of its low overhead and fast querying nature.
- **Online Games**: Many real-time multiplayer games use UDP to reduce latency, since timely data delivery is more important than guaranteed delivery.
- **VoIP (Voice over IP)**: Similar to streaming, VoIP applications use UDP to minimize delays in voice communication.

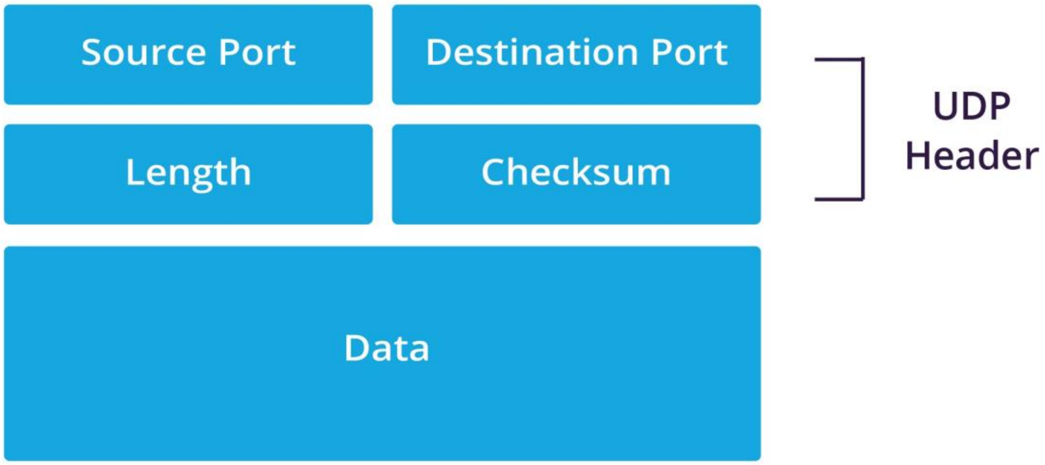
Transport Layer Headers



TCP OR UDP

Transport Layer
UDP Datagram

- The arrival and order of messages are not guaranteed.
- It doesn't guarantee transmission order.



Source Port | Destination Port

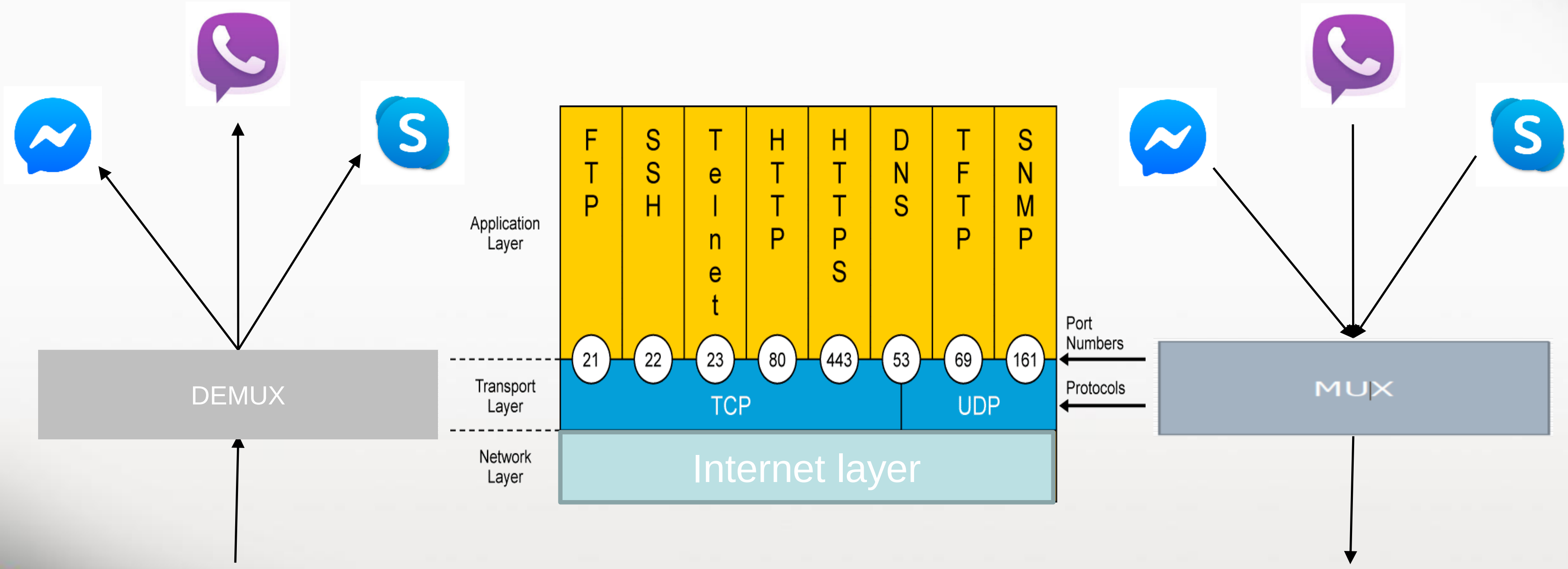
Length | Checksum

Data

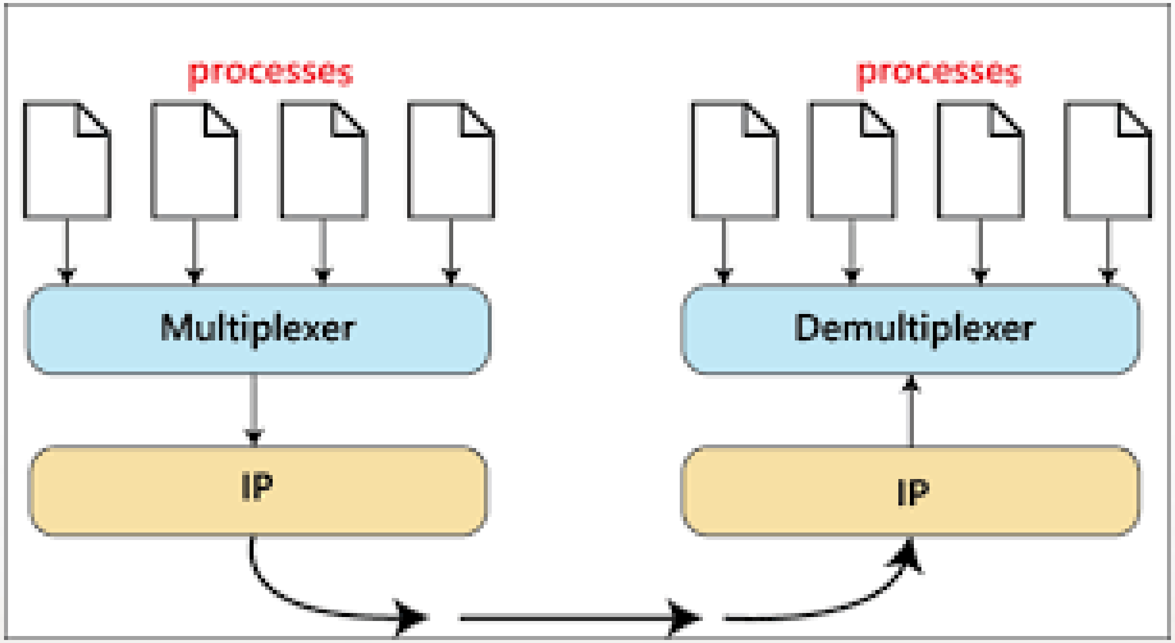
UDP Header

Transport Layer
UDP Multiplexing

UDP multiplexes and demultiplexes datagrams by selecting port numbers



Transport Layer Multiplexing



TCP Protocol
(Transfer Control Protocol)

TCP (Transmission Control Protocol) is one of the core protocols, responsible for ensuring reliable, ordered, and error-checked data transmission between computers in a network. It operates at the transport layer and is commonly paired with IP (Internet Protocol) to form the TCP/IP stack.

Key Features of TCP

Connection-Oriented: TCP establishes a reliable connection between the sender and receiver before transmitting data. This is done through a process called the three-way handshake, which ensures both parties are ready to communicate.

Reliable: TCP guarantees that all data sent from the sender will be received by the receiver in the same order, and without errors. If any data packets are lost or corrupted, TCP will automatically retransmit them.

Ordered Data Delivery: TCP ensures that data is delivered in the exact order in which it was sent. If packets arrive out of order, TCP will reorder them before passing them up to the application layer.

Flow Control: TCP uses a flow control mechanism called Windowing to manage the rate of data transmission between sender and receiver. This prevents network congestion and ensures that the receiver’s buffer does not overflow.

Key Features of TCP

Error Checking and Recovery: TCP includes error-checking mechanisms, such as checksums, to detect errors in transmitted data. If errors are found, the affected packets are discarded and retransmitted.

Congestion Control: TCP adjusts the rate of data transmission based on the current network conditions. If the network is congested, TCP reduces the rate at which data is sent, and if the network is clear, it increases the transmission rate.

Full-Duplex Communication: TCP allows data to be transmitted in both directions simultaneously. This means both the sender and receiver can send and receive data at the same time.

Common Use Cases:

Web Browsing (HTTP/HTTPS): Ensures that web pages load correctly with all resources intact.

File Transfer Protocol (FTP): Reliable transfer of files over the network.

Email Protocols (SMTP, IMAP, POP3): Reliable communication for email transmission and retrieval.

Remote Access (SSH, Telnet): Secure and reliable remote shell access.

TCP Comparison to UDP:

Reliability: TCP ensures reliable delivery, while UDP does not guarantee delivery or order.

Connection: TCP is connection-oriented, requiring a handshake to establish a connection, while UDP is connectionless.

Speed: TCP has more overhead because of error checking, flow control, and connection management, making it slower than UDP.

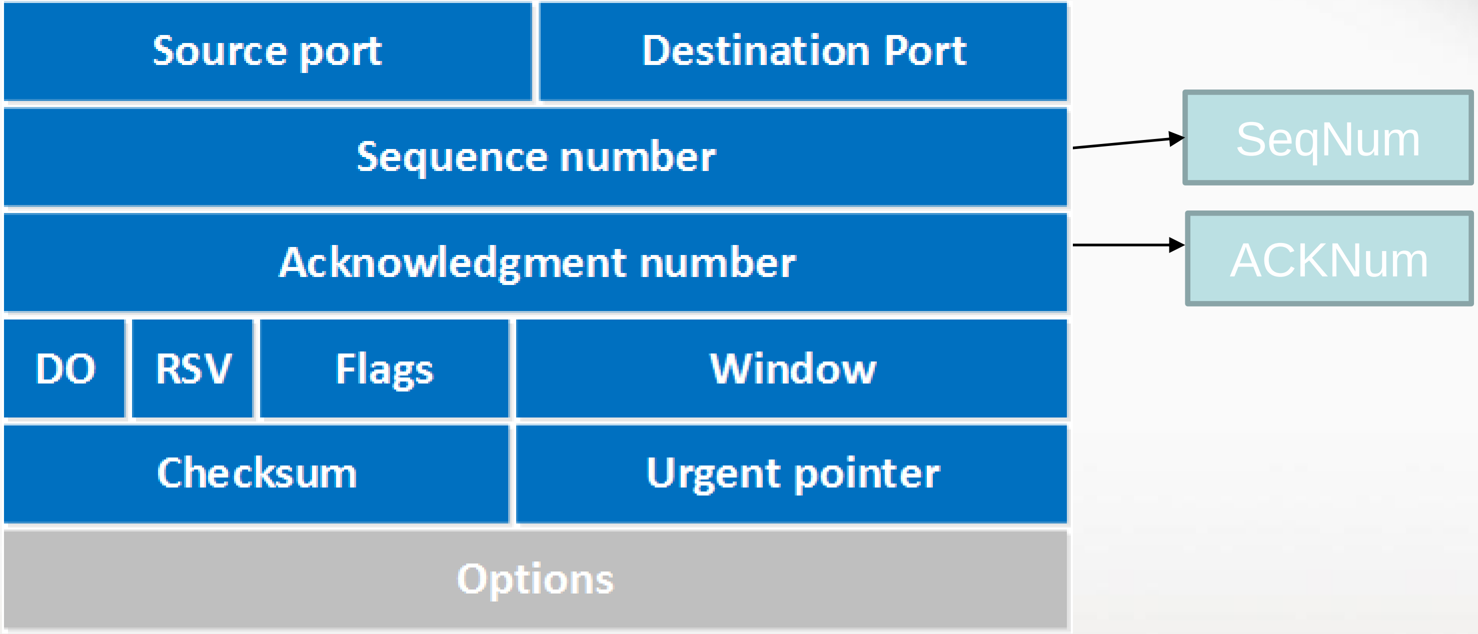
Use Cases: TCP is used for applications that require guaranteed data delivery, such as web browsing (HTTP/HTTPS), file transfers (FTP), email (SMTP), and remote access (SSH).

Transport Layer
TCP: Steps

The arrival and order of messages are guaranteed. Transmission order is ensured

1. Establish connections.
2. Transfer data.
3. Send acknowledgments.
4. Close or abort connections.

Transport Layer
TCP Segment



Source port

Destination Port

Sequence number

Acknowledgment number

DO

RSV

Flags

Window

Checksum

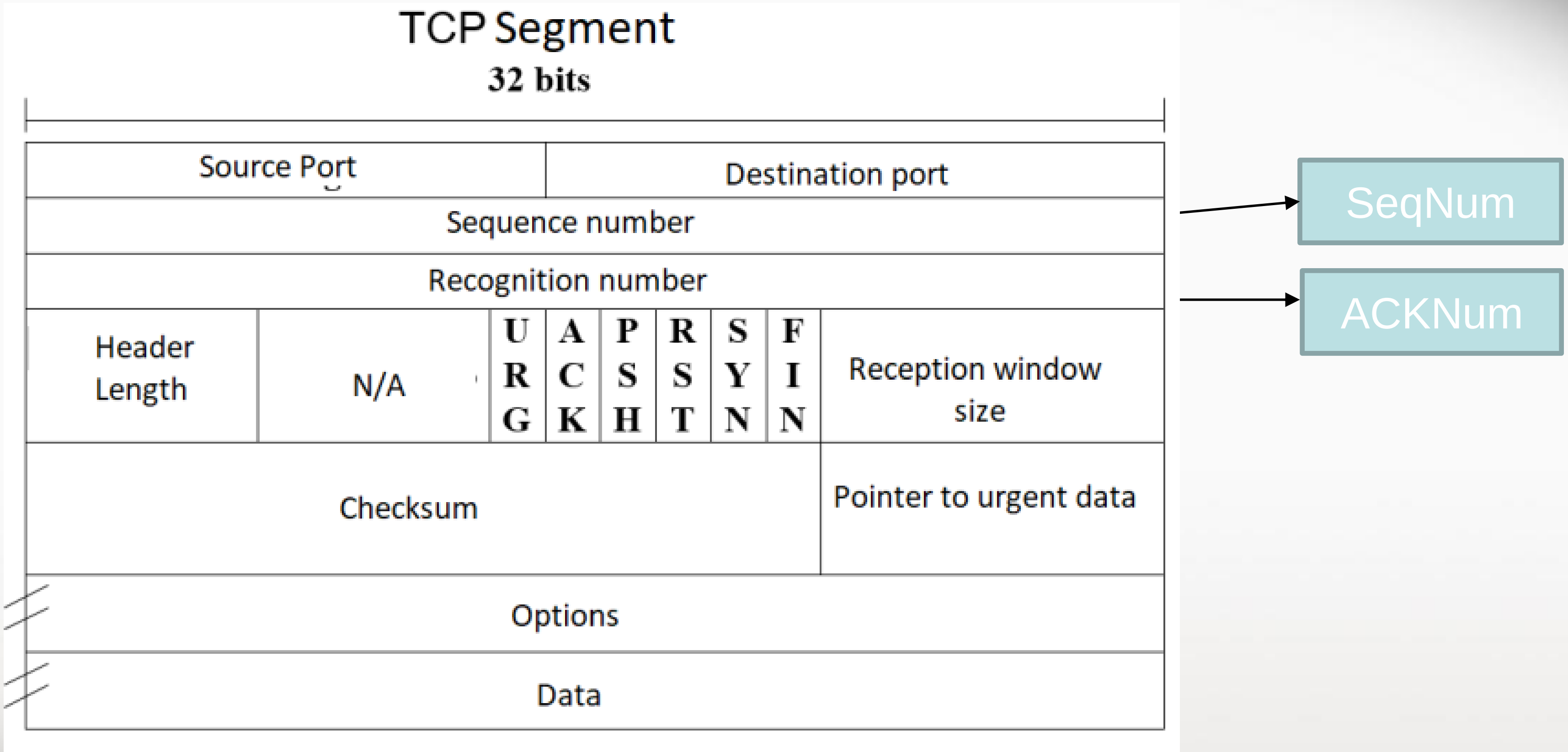
Urgent pointer

Options

SeqNum

ACKNum

Transport Layer
TCP Segment

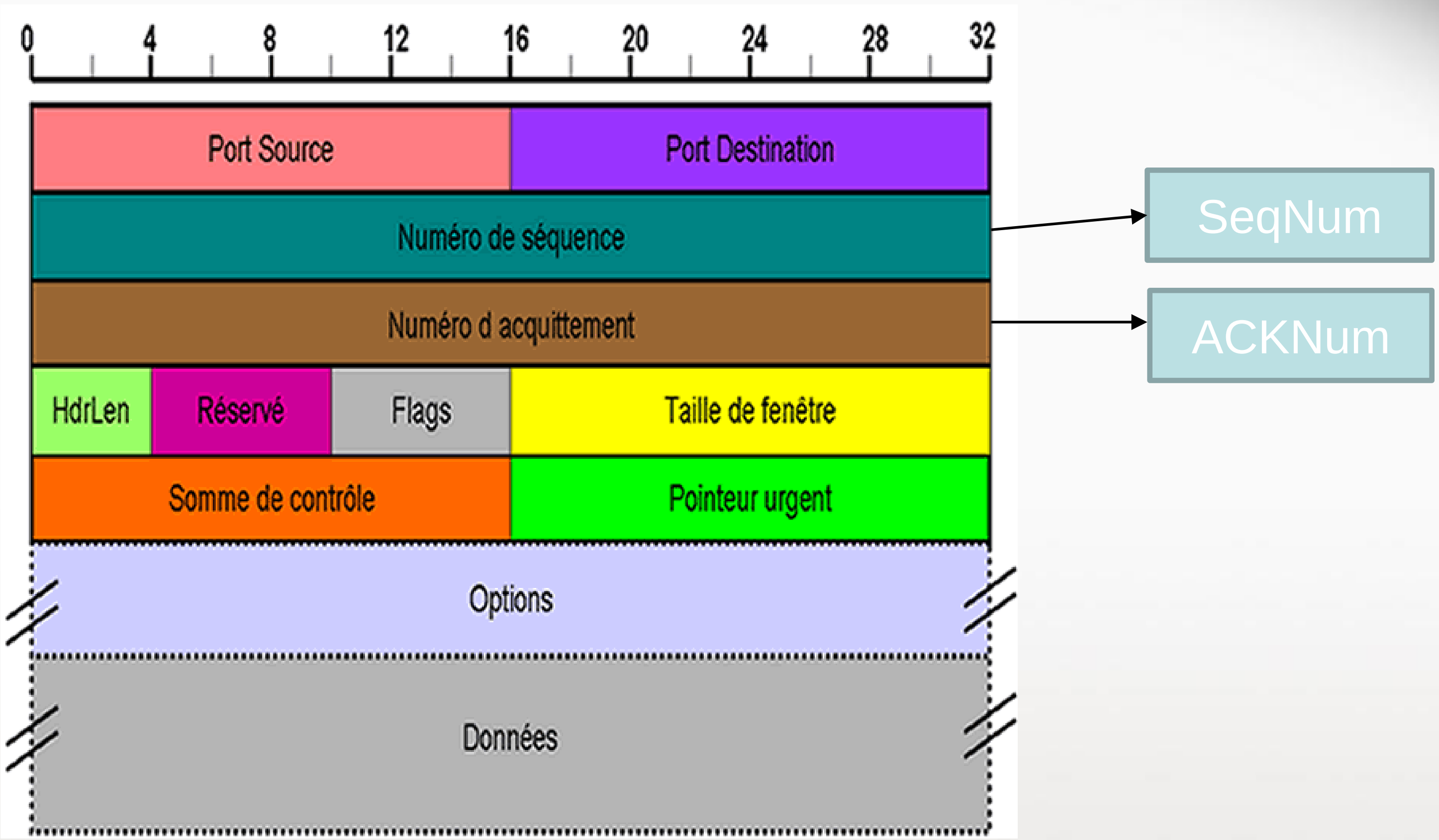


TCP Segment

32 bits

<table>
<tr><td colspan="3">Source Port</td><td colspan="6">Destination port</td></tr>
<tr><td colspan="9">Sequence number</td></tr>
<tr><td colspan="9">Recognition number</td></tr>
<tr><td>Header Length</td><td>N/A</td><td>URG</td><td>ACK</td><td>PSH</td><td>RST</td><td>SYN</td><td>FIN</td><td>Reception window size</td></tr>
<tr><td colspan="8">Checksum</td><td>Pointer to urgent data</td></tr>
<tr><td colspan="9">Options</td></tr>
<tr><td colspan="9">Data</td></tr>
</table>

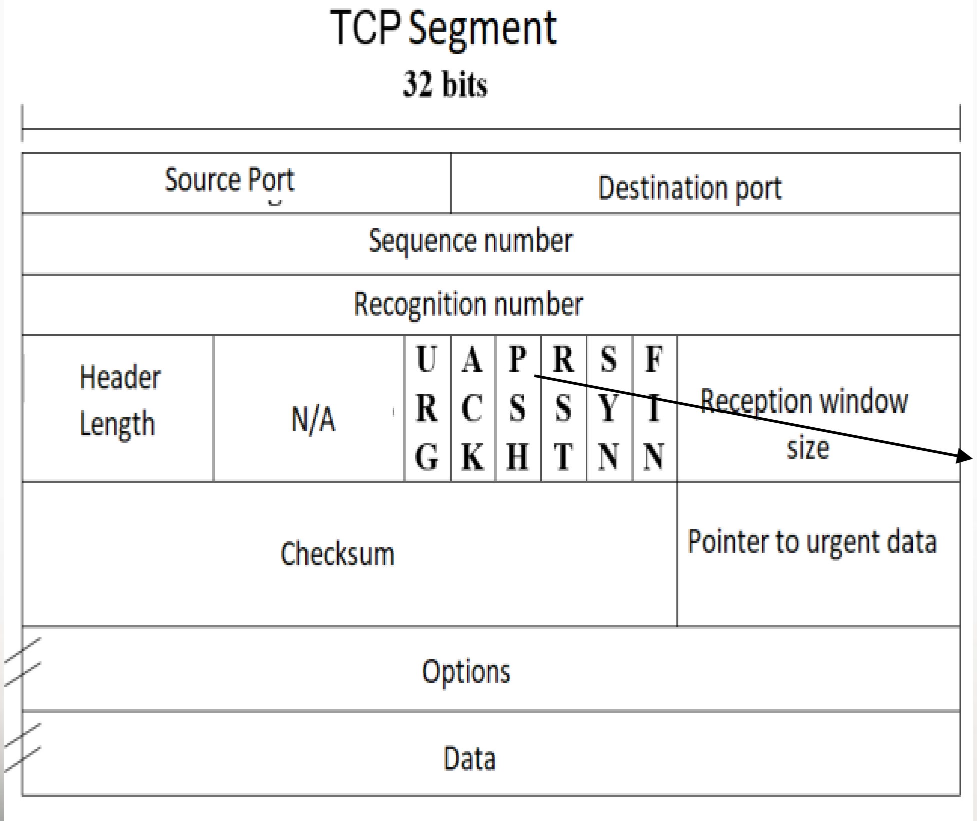
Transport Layer
TCP Segment



SeqNum

ACKNum

Couche Transport
TCP: Flags



TCP Segment

32 bits

Source Port | Destination port

Sequence number

Recognition number

Header Length | N/A | URG | ACK | PSH | RST | SYN | FIN | Reception window size

Checksum | Pointer to urgent data

Options

Data

ACK: The packet is an acknowledgment.
FIN: The sender has reached the end of its data stream.
RST: Reset the connection.
SYN: Synchronize the sequence numbers to initiate a connection.
PSH: Push function (indicates that data should be passed immediately to the application).
URG: Indicates the presence of urgent data.

0	1	1	1	0	1	0	1

TCP Packet Structure:

Source Port (2 bytes): The sending application's port number.

Destination Port (2 bytes): The receiving application's port number.

Sequence Number (4 bytes): A number that tracks the position of the data byte in the stream. It is used for ordering and acknowledgment.

Acknowledgment Number (4 bytes): If the ACK flag is set, this field contains the sequence number of the next byte that the sender expects to receive.

Data Offset (4 bits): Specifies where the data begins in the packet, as the header can vary in length.

TCP Packet Structure:

Flags (9 bits): Controls the state of the connection (e.g., SYN, ACK, FIN, RST).

Window Size (2 bytes): Specifies the size of the sender's receive window (used for flow control).

Checksum (2 bytes): Used for error-checking the header and data.

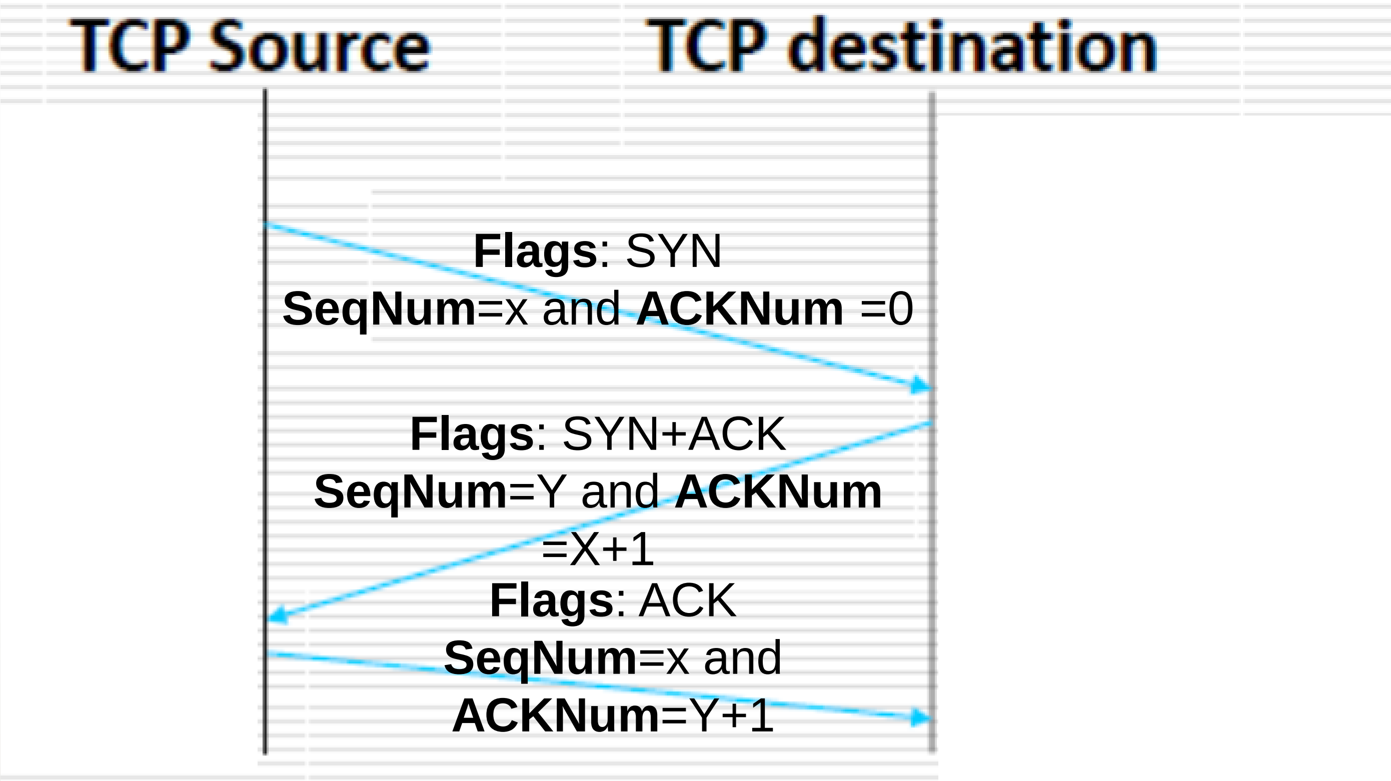
Urgent Pointer (2 bytes): Points to urgent data in the stream if the URG flag is set.

Options: Can include additional options for managing connections.

Transport Layer
TCP: Steps

1. Establish connections.
2. Transfer data.
3. Send acknowledgments.
4. Close or abort connections.

TCP: Establish connections three-way handshake

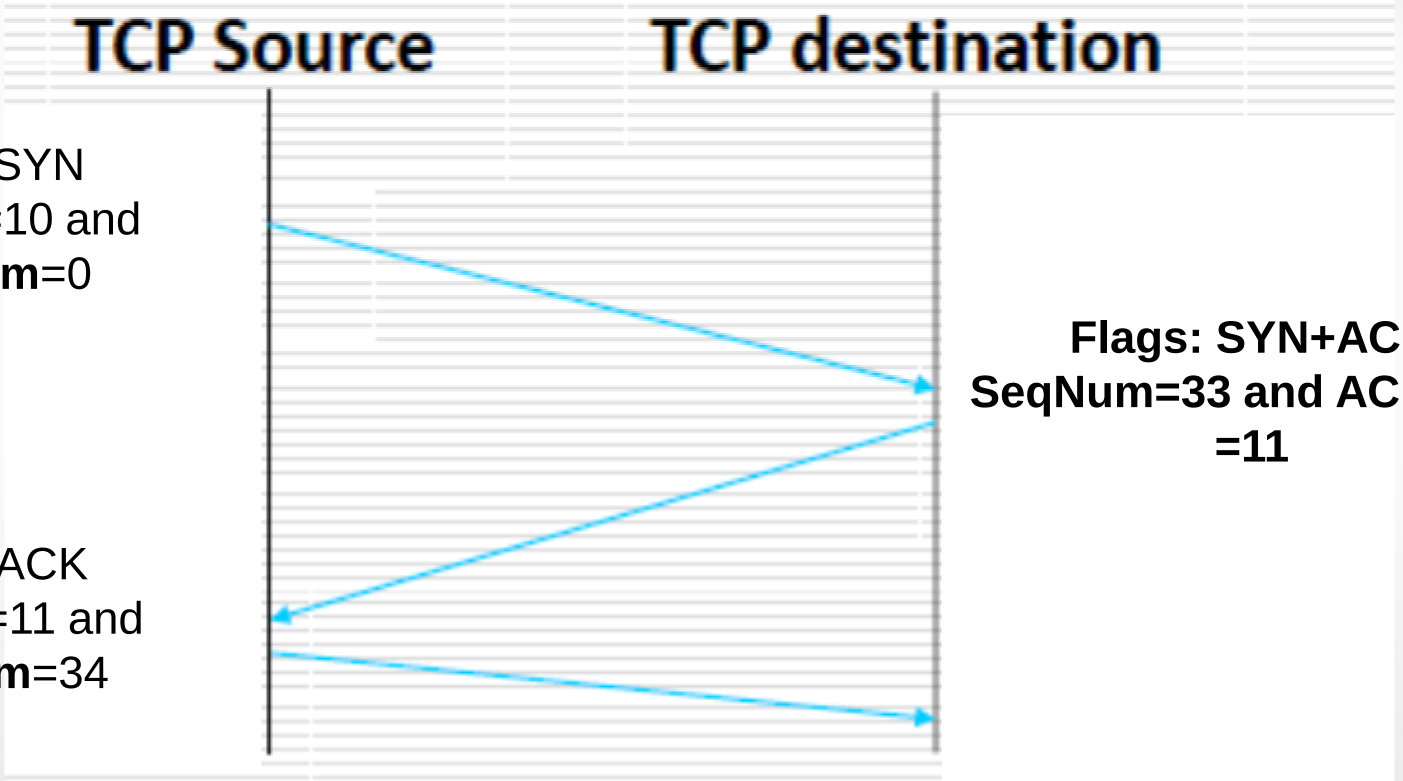


Flags: SYN
SeqNum=x and **ACKNum** =0

Flags: SYN+ACK
SeqNum=Y and **ACKNum** =X+1

Flags: ACK
SeqNum=x and **ACKNum**=Y+1

TCP: Establish connections

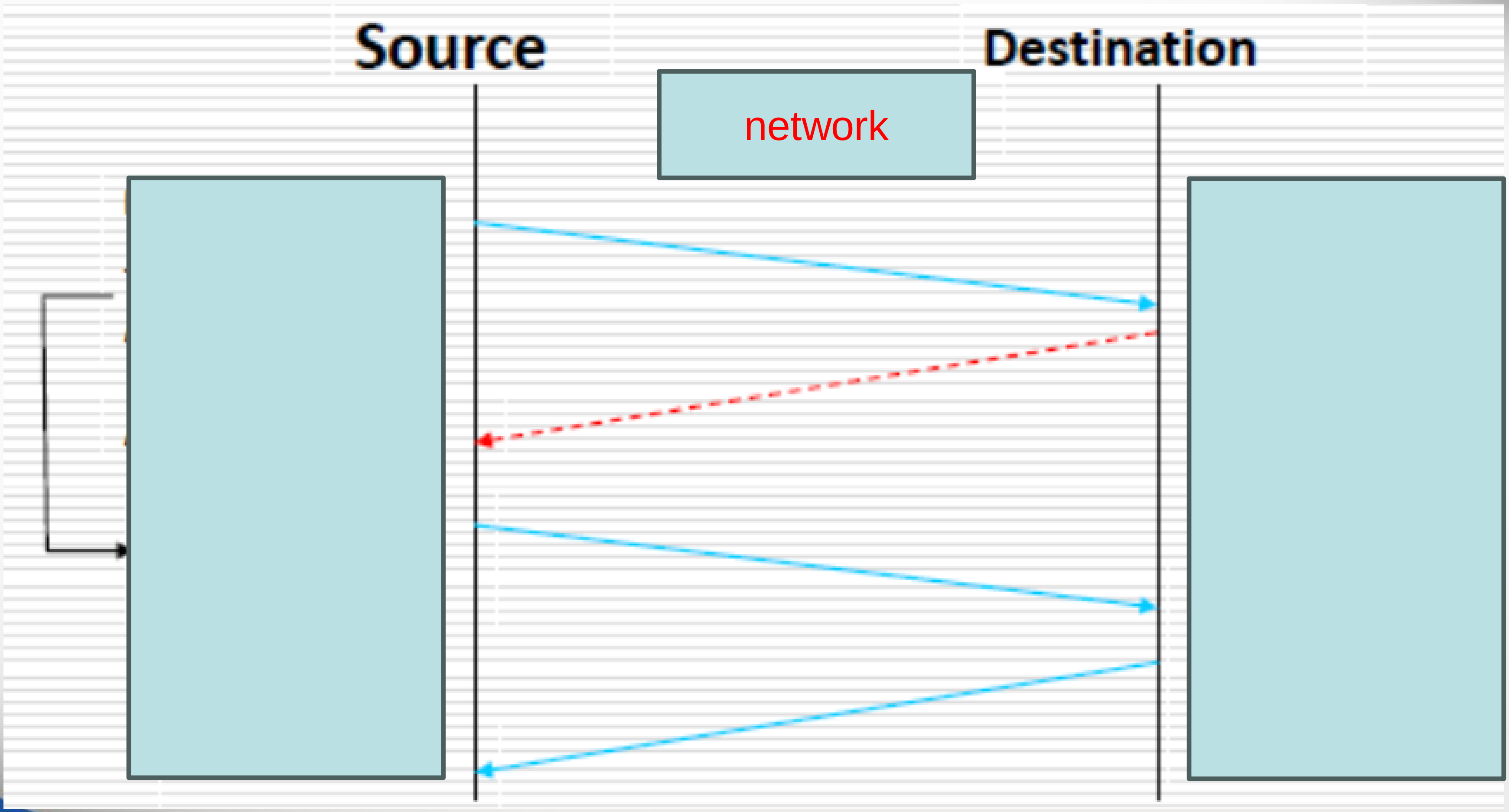


Flags: SYN
SeqNum=10 and
ACKNum=0

Flags: SYN+ACK
SeqNum=33 and ACKNum
=11

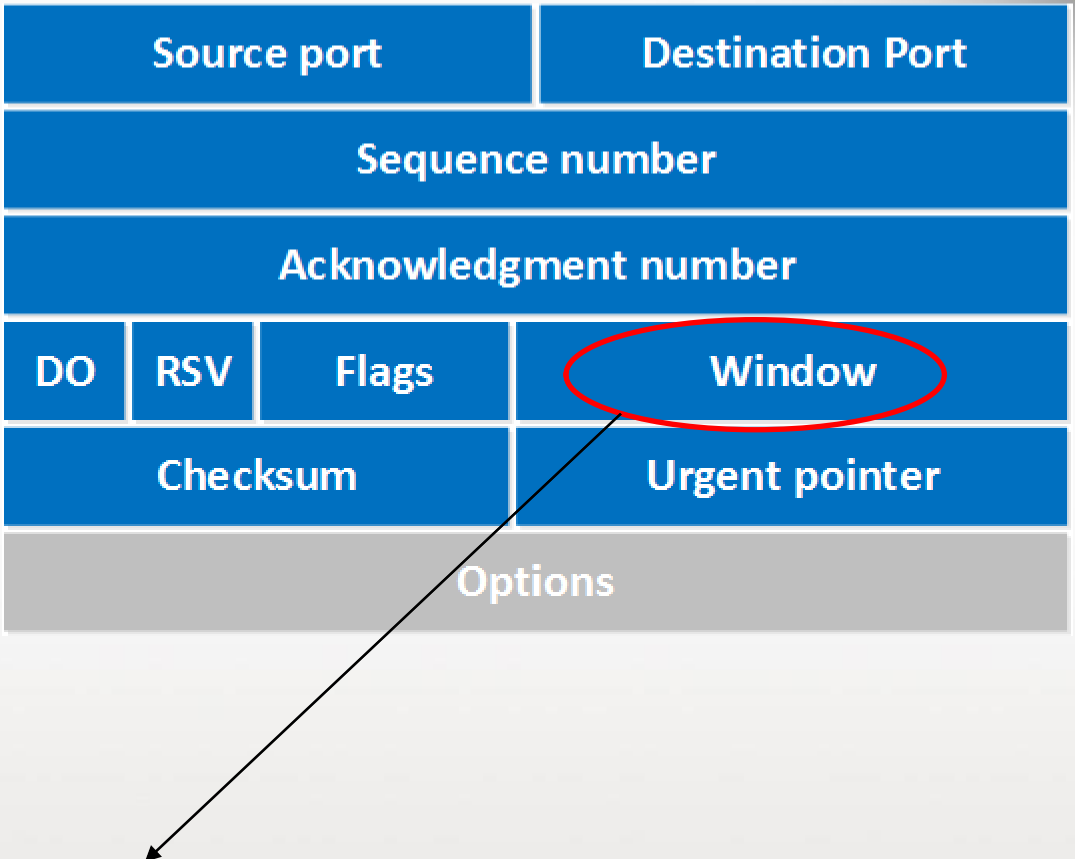
Flags: ACK
SeqNum=11 and
ACKNum=34

TCP: acknowledgments



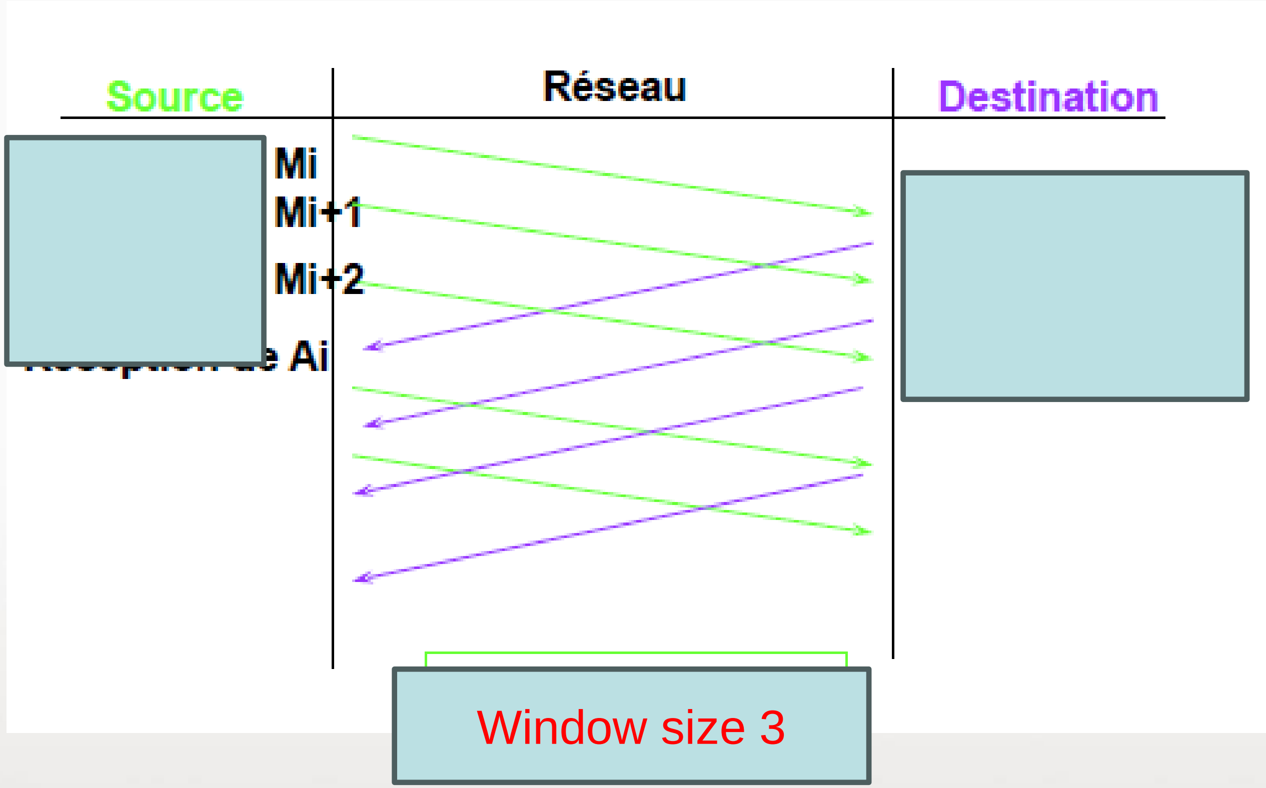
Transport Layer
TCP: Issue with Acknowledgments

- Bandwidth waste
  Sending data
  Waiting
  Sending acknowledgment
  Waiting
  .....



Get ahead of acknowledgments, i.e., sliding window

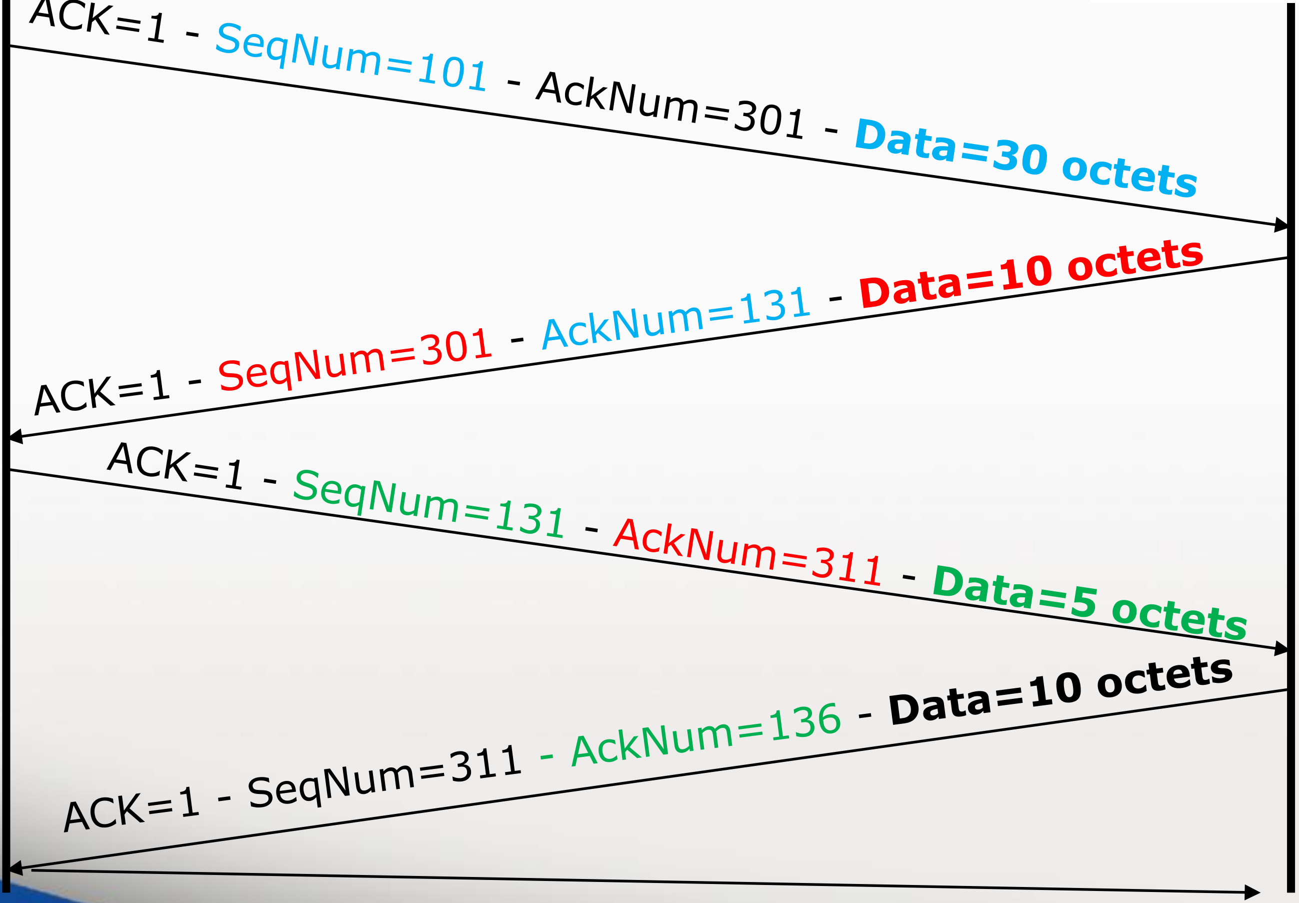
Transport Layer
TCP: Issue with Acknowledgments



Window size 3

Transport Layer
TCP: Data Transfer

ACKNum= SeqNum+Data



ACK=1 - SeqNum=101 - AckNum=301 - **Data=30 octets**

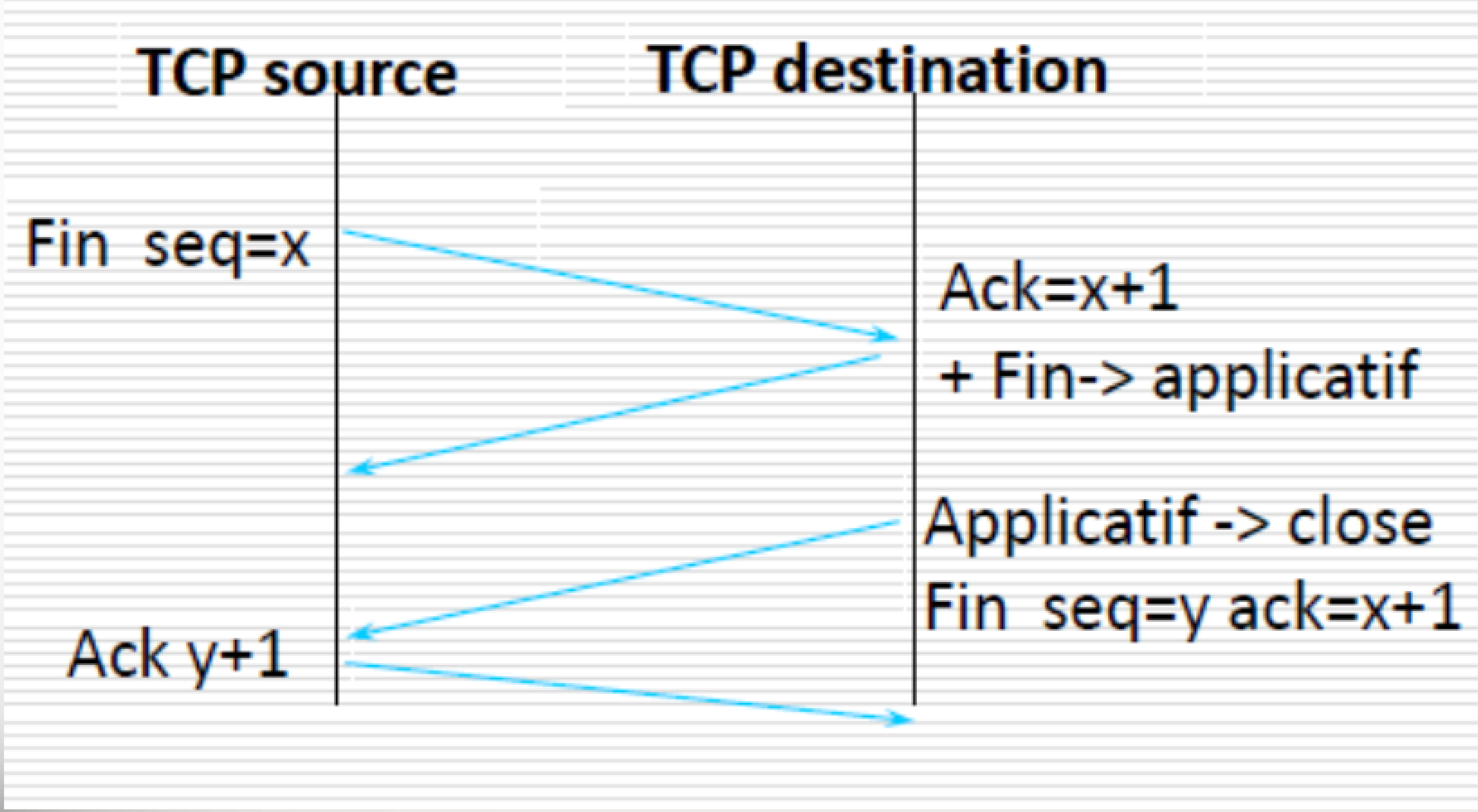
ACK=1 - SeqNum=301 - AckNum=131 - **Data=10 octets**

ACK=1 - SeqNum=131 - AckNum=311 - **Data=5 octets**

ACK=1 - SeqNum=311 - AckNum=136 - **Data=10 octets**

Réponse??? Exercice

Transport Layer
TCP: Connection Termination

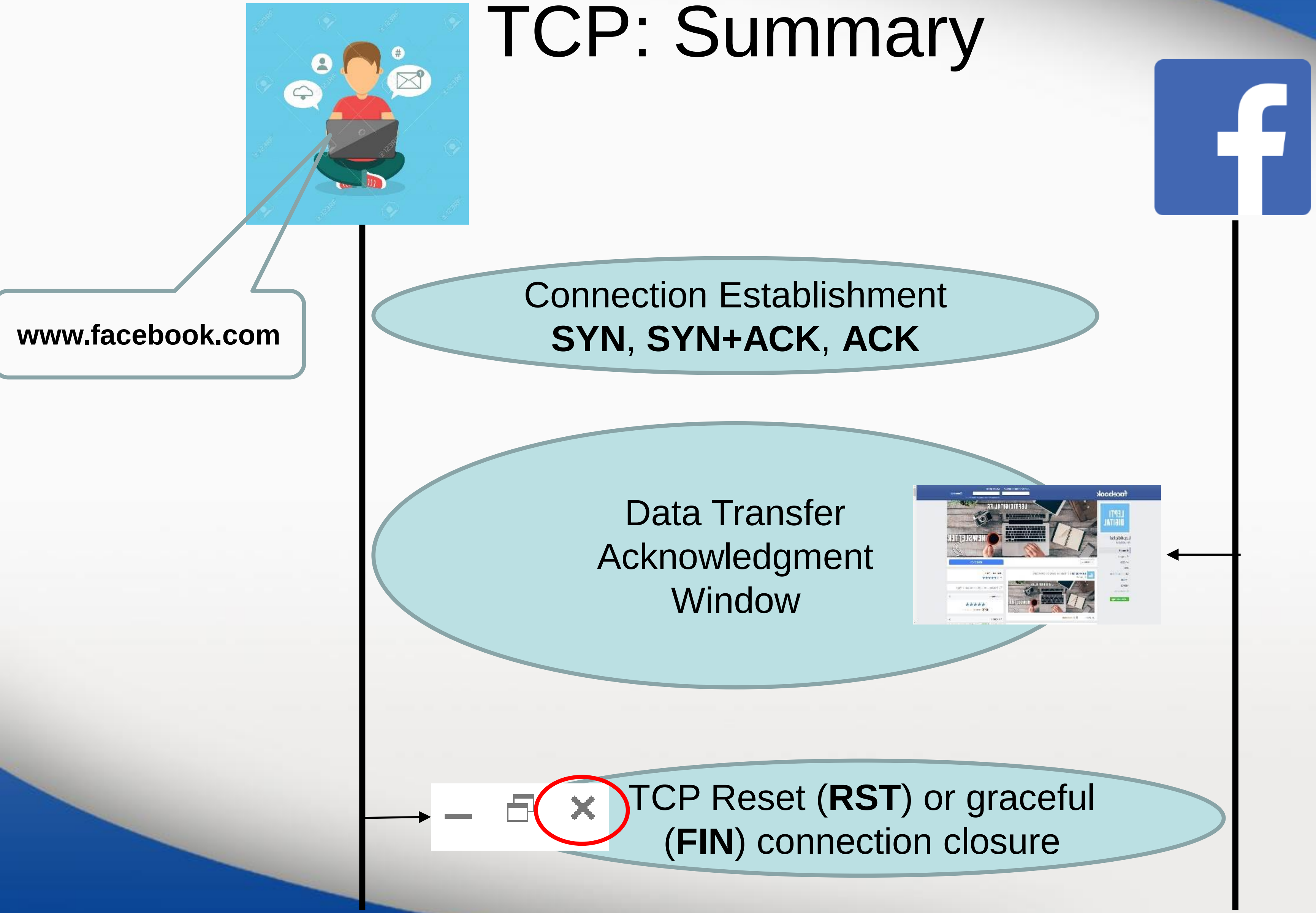


Transport Layer
TCP: TCP Reset

==> ACK=1 - RST=0 - SeqNum=200 - AckNum=400

<== ACK=0 - RST=1 - SeqNum=400 - ACKNum=xxx

Transport Layer
TCP: Summary



www.facebook.com

Connection Establishment
SYN, **SYN+ACK**, **ACK**

Data Transfer
Acknowledgment
Window

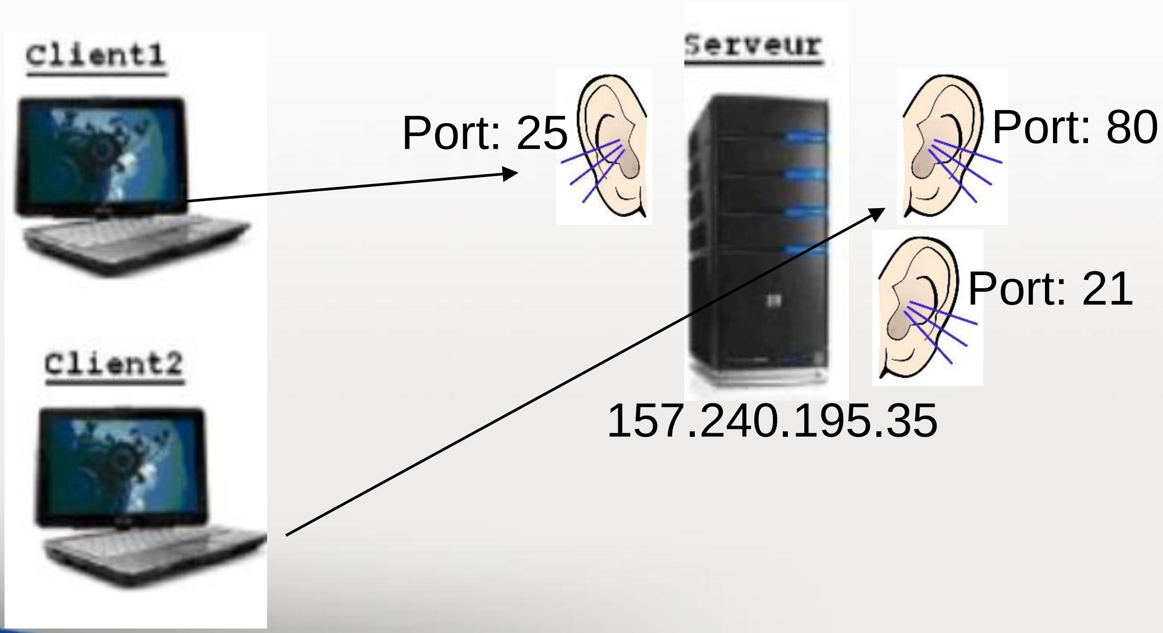
TCP Reset (**RST**) or graceful (**FIN**) connection closure

Client/Server Connection 3-tier

Transport Layer
Client/Server Connection

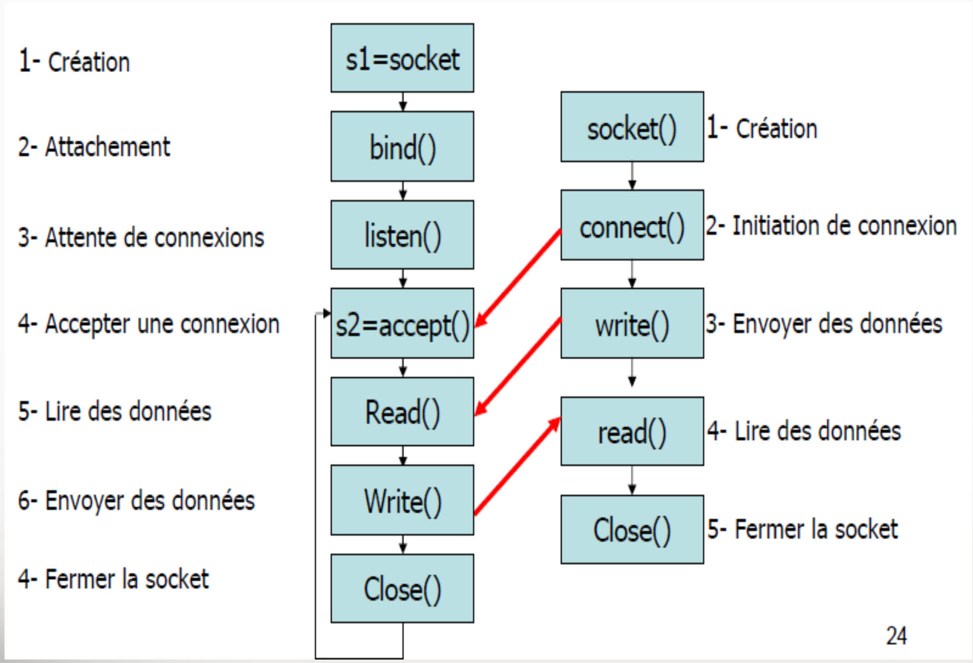
Socket = IP Address + Port Number (e.g., 200.24.32.11:80)

- The server initializes the IP address and port number of the socket on which it will listen for client requests.
- The client initializes the IP address and port number of the server's socket that it will contact.



Transport Layer
Client/Server Connection

Connection-
Oriented



1- Création
2- Attachement
3- Attente de connexions
4- Accepter une connexion
5- Lire des données
6- Envoyer des données
4- Fermer la socket

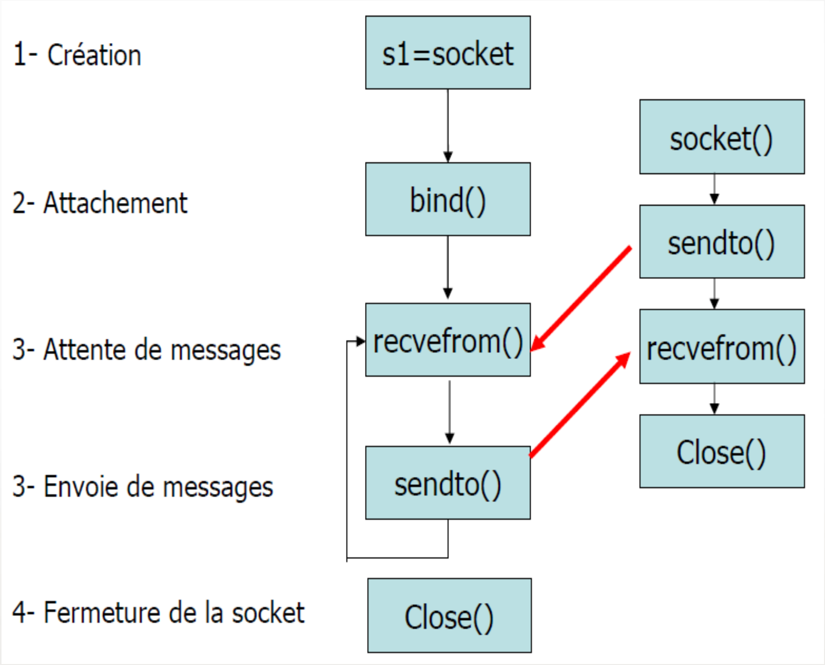
s1=socket
bind()
listen()
s2=accept()
Read()
Write()
Close()

socket() 1- Création
connect() 2- Initiation de connexion
write() 3- Envoyer des données
read() 4- Lire des données
Close() 5- Fermer la socket

24

Server Client

Connectionless



1- Création
2- Attachement
3- Attente de messages
3- Envoie de messages
4- Fermeture de la socket

s1=socket
bind()
recvefrom()
sendto()
Close()

socket()
sendto()
recvefrom()
Close()

Server Client

By Pr RIAHLA Mohamed Amine

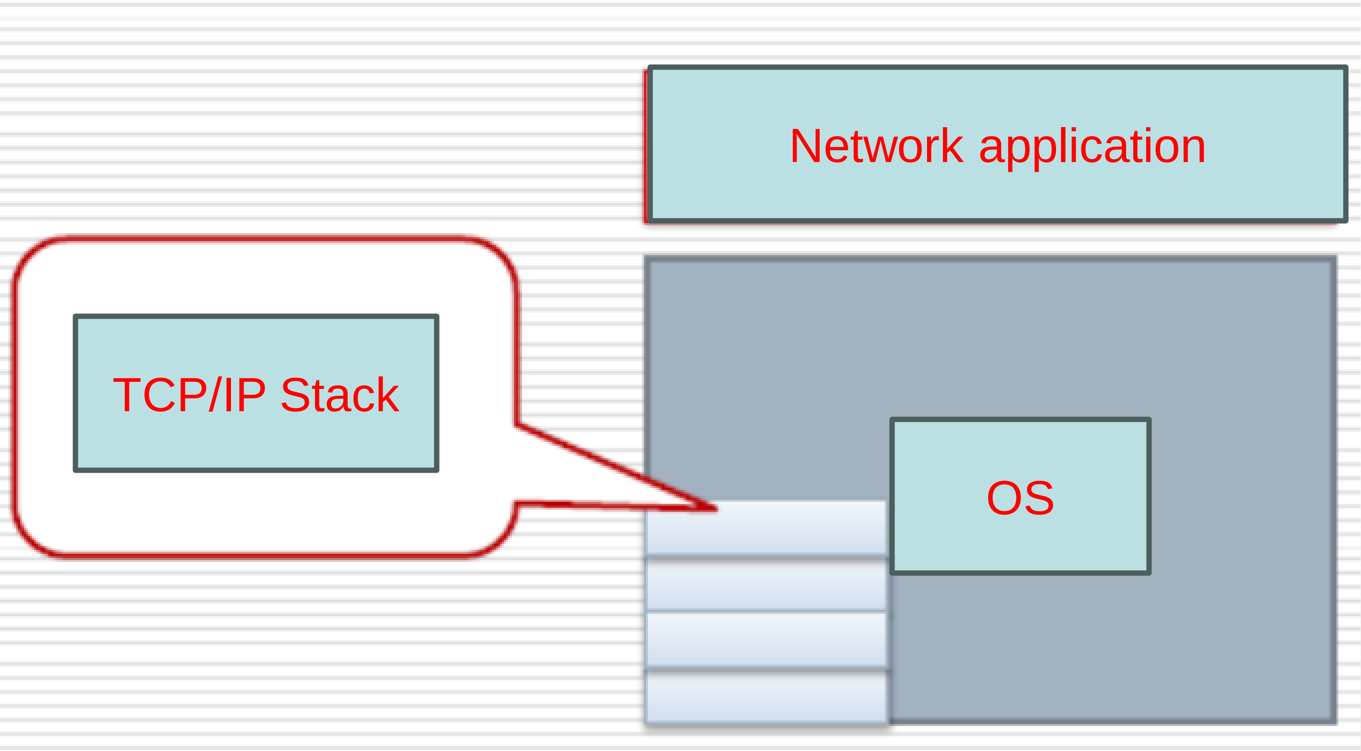
Transport Layer
Network Programming

By Pr RIAHLA Mohamed Amine

Transport Layer
Network Programming

By Pr RIAHLA Mohamed Amine

Transport Layer
Network Programming



Transport Layer Attacks

Denial of Service (DoS) Attacks

SYN Flood Attack:

•The attacker sends a large number of SYN requests (the first step in establishing a TCP connection) to a target server, but does not complete the handshake by sending the final ACK message.

•The server waits for a response, exhausting its resources (memory, processing power), which leads to denial of service for legitimate users.

UDP Flood:

•In this attack, the attacker sends large amounts of UDP packets to random ports on the target system. Since the system must process each UDP packet and respond, it can become overwhelmed, resulting in a denial of service.

Session Hijacking

Session Hijacking (TCP/IP Hijacking):

This attack occurs when an attacker takes control of a valid session between two parties by stealing or guessing session tokens or TCP sequence numbers.

Once the attacker has access to the session, they can impersonate the legitimate user, steal sensitive data, or disrupt communication. This is often done using packet sniffing and injecting malicious packets into the connection.

Other attacks

- **TCP Reset Attack (RST Attack):**The attacker sends a TCP reset (RST) packet to both sides of an active connection, causing the connection to be abruptly terminated

- **Traffic Analysis:**

- **Blind Injection Attacks:** the attacker injects data into an ongoing TCP session without receiving feedback from the system (hence "blind").

- **Port Scanning and Fingerprinting:** ex: NMAP

- **Amplification Attacks:** An attacker can send a small request to a vulnerable server that responds with a much larger response (DOS).

- **Application Layer Protocol Exploits (Through Transport Layer):** attackers might exploit HTTP floods (via TCP) or SMTP vulnerabilities to overwhelm a mail server with **spam** or malicious traffic.

- **TCP Congestion Attack:** In this attack, the attacker manipulates the congestion control mechanisms of TCP. The attacker may attempt to flood the network with packets, causing congestion and delays for legitimate users.

- **DDoS (Distributed Denial of Service) Attacks on the Transport Layer: botnets**

Prevention and Mitigation Techniques for Transport Layer Attacks

- **Firewalls and Intrusion Detection Systems (IDS):**Use firewalls to filter malicious traffic and IDS systems to detect unusual patterns that may indicate an attack.

- **SYN Cookies & Protection Against SYN Floods**: Implement SYN cookies to prevent SYN flood attacks. This method avoids allocating resources for half-open connections until the three-way handshake is complete.

- **Rate Limiting and Traffic Filtering**: Apply rate limiting to prevent brute-force attacks or excessive traffic, especially in UDP-based services.

- **Encryption (TLS/SSL) for TCP Connections**: Use encryption (e.g., SSL/TLS) to secure TCP sessions, preventing TCP hijacking and other man-in-the-middle attacks.

Prevention and Mitigation Techniques for Transport Layer Attacks

- **Strong Authentication and Session Management:** Use multi-factor authentication (MFA) and other strong session management techniques to prevent session hijacking**.**

- **DDoS Mitigation Services:** Use anti-DDoS services that can detect and mitigate DDoS attacks, often by distributing traffic across a large number of servers.

- **Deep Packet Inspection (DPI):** Implement DPI to inspect packets more deeply, identifying malicious activity that might be hidden within seemingly normal traffic.

- **Keep Systems and Protocols Updated:** Regularly patch vulnerabilities in systems and protocols, especially for services running over the Transport Layer.